USPS EXPRESS MAIL MAILING LABEL NO. EL 969195266 US

TITLE OF THE INVENTION

First Computer Process and Second Computer Process Proxy-Executing Code on Behalf Thereof

TECHNICAL FIELD

[0001] This invention relates to a first process on a computer and a second process on the computer that executes code on behalf of and as a proxy for such first process. More particularly, the invention relates to a security process on the computer that proxy-executes the code on behalf of an application process only if the security process is satisfied based on a license or the like that the application process is entitled to be operating on the computer.

BACKGROUND OF THE INVENTION

[0002] A computer application distributor wishes to distribute such computer application to each of many users or recipients in exchange for a license fee or some other consideration. However, such distributor typically also wishes to restrict what each user or recipient can do with such distributed computer application. For example, the distributor would like to restrict the user from

copying and re-distributing such application to a second user, at least in a manner that denies the distributor a license fee from such second user.

[0003] In addition, the distributor may wish to provide the user with the flexibility to purchase different types of use licenses at different license fees, while at the same time holding the user to the terms of whatever type of license is in fact purchased. For example, the distributor may wish to allow the application to be executed only a limited number of times, only for a certain total time, only on a certain type of machine, only on a certain type of rendering platform, only by a certain type of user, etc. Likewise, the distributor may wish to allow one user to pay a smaller license fee and access a smaller set of application functions and also to allow another user to pay a larger license fee and access a larger set of application functions, and the like.

[0004] However, after distribution has occurred, such distributor has very little if any control over the distributed application. This is especially problematic in view of the fact that the application may be copied and re-distributed to most any personal computer, presuming that the application is not otherwise protected in some manner from such copying and re-distribution. As should be appreciated, most any such personal computer includes the software and hardware necessary to make an exact digital copy of such application, and to download such exact digital copy to a write-able magnetic or optical disk, or to send such exact digital copy over a network such as the Internet to any destination.

[0005] Of course, as part of a transaction wherein the application is distributed, the distributor may require the user / recipient of the application to promise not to re-distribute such application in an unwelcome manner. However, such a promise is easily made and easily broken. A distributor may therefore attempt to prevent such re-distribution through any of several known security measures.

[0006] One such security measure is product activation. In such product activation, a customer acquiring a software application is provided with a product activation key corresponding thereto, which is a unique serial number and product identifier that acts as a proof of purchase or the like. The provided product key is then entered during installation of the application on a particular computer device to act as a proffer that the application was acquired legally and/or otherwise

properly. The product activation key need not be and typically is not cryptographic in nature, although a digital signature (which is cryptographic in nature) may be included to act as a guarantee that the product key is genuine.

[0007] The entered product key and an ID representative of the computer device are then sent to a product activation service as part of the installation process. As may be appreciated, the product activation service determines whether the entered product key is valid, whether the product key has been employed before, and if so in connection with what computer device. Typically, each product key enables an installation or re-installation of the application on a single computing device, as is set forth in a corresponding license agreement, although a product key may also enable a set number of installations / re-installations on multiple computer devices also.

[0008] Accordingly, if the product activation service determines that the entered product key has already been employed to install the application on another computer device (or has been employed a maximum number of times, for example), such activation service will not allow the installation of the application on the computer device to proceed, will not allow a complete installation of the application on the computer device, will not allow the installed application to be used on the computer device, or the like, as the case maybe. Thus, activation as used herein may entail permission to install the application, permission to perform some level of installation of the application, permission to completely install the application, some level of permission to use the application, complete permission to use the application, or the like.

[0009] If the activation service declines to activate the application for the customer based on an entered product key already being used in connection with another computing device, or based on the entered product key not supporting the level of activation desired, the customer must acquire another appropriate product key to install / completely install / use the application on the computing device in the manner desired. Thus, the product key and the product activation service act to ensure that the application is not nefariously or wantonly installed / activated / used on multiple computing devices, such as may be in violation of any software license agreement associated with the software product.

[0010] Note that as part of the activation process, the activation service may return a digital version of the license to the computing device on which the application is associated. Such license may be tied to the computing device such that the license is not usable with any other computing device, and may express a level of activation, as well as license terms such as application functions that are to be made available, functions that are to be made non-available, a period of activation or a number of times the application may be executed on the computing device, and the like. In general, such license may express any limitations and/or rights and also may express any policies that should be honored in connection with the execution of the application on the computing device, all as set forth by the distributor of the application or another entity.

[0011] With such license, then, a rights client controller with a license evaluator or the like may be employed on the computer along with the distributed application to control operation and use of the application based on an evaluation of whether the license so permits. However, a need exists for an actual method and mechanism by which such rights client with such license evaluator may in fact control operation and use of the application based on the license. In particular, a need exists for such a rights client with such a license evaluator that executes certain portions of code on behalf of and as a proxy for the application, but only if the license evaluator determines that the license allows such execution.

SUMMARY OF THE INVENTION

[0012] The aforementioned needs are satisfied at least in part by the present invention in which a computer has thereon a first process operating on the computer comprising code to be executed in connection therewith, where the code includes at least one triggering device, and a digital license corresponding to the first process, where the license sets forth terms and conditions for operating the first process. A second process operating on the computer proxy-executes code corresponding to each triggering device of the first process on behalf of such first process. The second process includes a license evaluator for evaluating the license to determine whether the first process is to be operated in accordance with the terms and conditions set forth in such license, and the second process

chooses whether to in fact proxy-execute the code corresponding to each triggering device of the first process on behalf of such first process based at least in part on whether the license evaluator has determined that the first process is to be operated in accordance with the terms and conditions of the license. Thus, the first process is dependent upon the second process for operation thereof.

[0013] The second process monitors for when the first process executes a triggering device thereof. Upon such occurrence, the second process responds thereto by determining an address of the triggering device within the first process, locating in a table the code section corresponding to the triggering device based on the determined address, and proxy-executing the located code section on behalf of the first process.

[0014] To develop the first process, source code is developed in an appropriate programming language, and within such source code each of one or more code sections that is to be proxy-executed by the second process is identified. The source code is compiled into machine code such that an identification of each identified code section is maintained, and the machine code is post-compiled with each identified code section therein into final code representative of the first process based on the identification of each identified code section by for each identified code section converting same into a form accessible only by the second process and not by the first process.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] The foregoing summary, as well as the following detailed description of the embodiments of the present invention, will be better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there are shown in the drawings embodiments which are presently preferred. As should be understood, however, the invention is not limited to the precise arrangements and instrumentalities shown. In the drawings:

[0016] Fig. 1 is a block diagram representing an exemplary non-limiting computing environment in which the present invention may be implemented;

[0017] Fig. 2 is a block diagram representing an exemplary network environment having a variety of computing devices in which the present invention may be implemented;

[0018] Fig. 3 is a block diagram showing a first computer process, a second computer process proxy-executing code on behalf of the computer process, and related elements in accordance with one embodiment of the present invention;

[0019] Fig. 4 is a flow diagram showing key steps performed in connection with the first and second processes of Fig. 3 to proxy-execute code in accordance with one embodiment of the present invention; and

[0020] Fig. 5 is a flow diagram showing key steps performed to develop the first process of Fig. 3 in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

COMPUTER ENVIRONMENT

[0021] Fig. 1 and the following discussion are intended to provide a brief general description of a suitable computing environment in which the invention may be implemented. It should be understood, however, that handheld, portable, and other computing devices of all kinds are contemplated for use in connection with the present invention. While a general purpose computer is described below, this is but one example, and the present invention requires only a thin client having network server interoperability and interaction. Thus, the present invention may be implemented in an environment of networked hosted services in which very little or minimal client resources are implicated, e.g., a networked environment in which the client device serves merely as a browser or interface to the World Wide Web.

[0022] Although not required, the invention can be implemented via an application programming interface (API), for use by a developer, and/or included within the network browsing software which will be described in the general context of computer-executable instructions, such as program modules, being executed by one or more computers, such as client workstations, servers, or other devices. Generally, program modules include routines, programs, objects,

components, data structures and the like that perform particular tasks or implement particular abstract data types. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations. Other well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers (PCs), automated teller machines, server computers, hand-held or laptop devices, multi-processor systems, microprocessor-based systems, programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network or other data transmission medium. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[0023] Fig. 1 thus illustrates an example of a suitable computing system environment 100 in which the invention may be implemented, although as made clear above, the computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0024] With reference to Fig. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel

Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus (also known as Mezzanine bus).

[0025] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CDROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared, and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[0026] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately

accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Fig. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0027] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Fig. 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156, such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0028] The drives and their associated computer storage media discussed above and illustrated in Fig. 1 provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In Fig. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 110 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is

coupled to the system bus 121, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB).

[0029] A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. A graphics interface 182, such as Northbridge, may also be connected to the system bus 121. Northbridge is a chipset that communicates with the CPU, or host processing unit 120, and assumes responsibility for accelerated graphics port (AGP) communications. One or more graphics processing units (GPUs) 184 may communicate with graphics interface 182. In this regard, GPUs 184 generally include on-chip memory storage, such as register storage and GPUs 184 communicate with a video memory 186. GPUs 184, however, are but one example of a coprocessor and thus a variety of co-processing devices may be included in computer 110. A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190, which may in turn communicate with video memory 186. In addition to monitor 191, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195.

[0030] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in Fig. 1. The logical connections depicted in Fig. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0031] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173,

such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Fig. 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0032] One of ordinary skill in the art can appreciate that a computer 110 or other client device can be deployed as part of a computer network. In this regard, the present invention pertains to any computer system having any number of memory or storage units, and any number of applications and processes occurring across any number of storage units or volumes. The present invention may apply to an environment with server computers and client computers deployed in a network environment, having remote or local storage. The present invention may also apply to a standalone computing device, having programming language functionality, interpretation and execution capabilities.

[0033] Distributed computing facilitates sharing of computer resources and services by direct exchange between computing devices and systems. These resources and services include the exchange of information, cache storage, and disk storage for files. Distributed computing takes advantage of network connectivity, allowing clients to leverage their collective power to benefit the entire enterprise. In this regard, a variety of devices may have applications, objects or resources that may interact to implicate authentication techniques of the present invention for trusted graphics pipeline(s).

[0034] Fig. 2 provides a schematic diagram of an exemplary networked or distributed computing environment. The distributed computing environment comprises computing objects 10a, 10b, etc. and computing objects or devices 110a, 110b, 110c, etc. These objects may comprise programs, methods, data stores, programmable logic, etc. The objects may comprise portions of the same or different devices such as PDAs, televisions, MP3 players, televisions, personal computers, etc. Each object can communicate with another object by way of the

communications network 14. This network may itself comprise other computing objects and computing devices that provide services to the system of Fig. 2. In accordance with an aspect of the invention, each object 10 or 110 may contain an application that might request the authentication techniques of the present invention for trusted graphics pipeline(s).

[0035] It can also be appreciated that an object, such as 110c, may be hosted on another computing device 10 or 110. Thus, although the physical environment depicted may show the connected devices as computers, such illustration is merely exemplary and the physical environment may alternatively be depicted or described comprising various digital devices such as PDAs, televisions, MP3 players, etc., software objects such as interfaces, COM objects and the like.

[0036] There are a variety of systems, components, and network configurations that support distributed computing environments. For example, computing systems may be connected together by wireline or wireless systems, by local networks or widely distributed networks. Currently, many of the networks are coupled to the Internet, which provides the infrastructure for widely distributed computing and encompasses many different networks.

[0037] In home networking environments, there are at least four disparate network transport media that may each support a unique protocol such as Power line, data (both wireless and wired), voice (e.g., telephone) and entertainment media. Most home control devices such as light switches and appliances may use power line for connectivity. Data Services may enter the home as broadband (e.g., either DSL or Cable modem) and are accessible within the home using either wireless (e.g., HomeRF or 802.11b) or wired (e.g., Home PNA, Cat 5, even power line) connectivity. Voice traffic may enter the home either as wired (e.g., Cat 3) or wireless (e.g., cell phones) and may be distributed within the home using Cat 3 wiring. Entertainment media may enter the home either through satellite or cable and is typically distributed in the home using coaxial cable. IEEE 1394 and DVI are also emerging as digital interconnects for clusters of media devices. All of these network environments and others that may emerge as protocol standards may be interconnected to form an intranet that may be connected to the outside world by way of the Internet. In short, a variety of disparate sources exist for the

storage and transmission of data, and consequently, moving forward, computing devices will require ways of protecting content at all portions of the data processing pipeline.

[0038] The 'Internet' commonly refers to the collection of networks and gateways that utilize the TCP/IP suite of protocols, which are well-known in the art of computer networking. TCP/IP is an acronym for "Transport Control Protocol/Interface Program." The Internet can be described as a system of geographically distributed remote computer networks interconnected by computers executing networking protocols that allow users to interact and share information over the networks. Because of such wide-spread information sharing, remote networks such as the Internet have thus far generally evolved into an open system for which developers can design software applications for performing specialized operations or services, essentially without restriction.

[0039] Thus, the network infrastructure enables a host of network topologies such as client/server, peer-to-peer, or hybrid architectures. The "client" is a member of a class or group that uses the services of another class or group to which it is not related. Thus, in computing, a client is a process, i.e., roughly a set of instructions or tasks, that requests a service provided by another program. The client process utilizes the requested service without having to "know" any working details about the other program or the service itself. In a client/server architecture, particularly a networked system, a client is usually a computer that accesses shared network resources provided by another computer e.g., a server. In the example of Fig. 2, computers 110a, 110b, etc. can be thought of as clients and computer 10a, 10b, etc. can be thought of as the server where server 10a, 10b, etc. maintains the data that is then replicated in the client computers 110a, 110b, etc.

[0040] A server is typically a remote computer system accessible over a remote network such as the Internet. The client process may be active in a first computer system, and the server process may be active in a second computer system, communicating with one another over a communications medium, thus providing distributed functionality and allowing multiple clients to take advantage of the information-gathering capabilities of the server.

[0041] Client and server communicate with one another utilizing the functionality provided by a protocol layer. For example, Hypertext-Transfer Protocol (HTTP) is a common protocol that is used in conjunction with the World Wide Web (WWW). Typically, a computer network address such as a Universal Resource Locator (URL) or an Internet Protocol (IP) address is used to identify the server or client computers to each other. The network address can be referred to as a Universal Resource Locator address. For example, communication can be provided over a communications medium. In particular, the client and server may be coupled to one another via TCP/IP connections for high-capacity communication.

[0042] Thus, Fig. 2 illustrates an exemplary networked or distributed environment, with a server in communication with client computers via a network/bus, in which the present invention may be employed. In more detail, a number of servers 10a, 10b, etc., are interconnected via a communications network/bus 14, which may be a LAN, WAN, intranet, the Internet, etc., with a number of client or remote computing devices 110a, 110b, 110c, 110d, 110e, etc., such as a portable computer, handheld computer, thin client, networked appliance, or other device, such as a VCR, TV, oven, light, heater and the like in accordance with the present invention. It is thus contemplated that the present invention may apply to any computing device in connection with which it is desirable to process, store or render secure content from a trusted source.

[0043] In a network environment in which the communications network/bus 14 is the Internet, for example, the servers 10 can be Web servers with which the clients 110a, 110b, 110c, 110d, 110e, etc. communicate via any of a number of known protocols such as HTTP. Servers 10 may also serve as clients 110, as may be characteristic of a distributed computing environment. Communications may be wired or wireless, where appropriate. Client devices 110 may or may not communicate via communications network/bus 14, and may have independent communications associated therewith. For example, in the case of a TV or VCR, there may or may not be a networked aspect to the control thereof. Each client computer 110 and server computer 10 may be equipped with various application program modules or objects 135 and with connections or access to various types of storage elements or objects, across which files may be stored or to which

portion(s) of files may be downloaded or migrated. Thus, the present invention can be utilized in a computer network environment having client computers 110a, 110b, etc. that can access and interact with a computer network/bus 14 and server computers 10a, 10b, etc. that may interact with client computers 110a, 110b, etc. and other devices 111 and databases 20.

Proxy Execution of Code

[0044] In the present invention, a rights client with a license evaluator and in connection with a product activation service controls operation and use of an application based on a corresponding license by executing code on behalf of and as a proxy for an application, but only if the license evaluator determines that the license allows such execution. Thus, the rights client with the license evaluator enforces the license as against a user of the application.

[0045] As may be appreciated, although the present invention is disclosed primarily in terms of the rights client with the license evaluator, the application, the license, and the product activation service, such present invention may also be employed in connection with alternate elements without departing from the spirit and scope of the present invention. For example, the application may instead be any application or type of process running on a computer, including a program, an operating system, and the like, or even a piece of digital content such as an audio recording or multimedia presentation. Similarly, the license may instead be any sort of permission token, with or without specific permission parameters, and the license evaluator may instead be any kind of device for evaluating such a permission token. Likewise, the product activation service may instead be any variety of permission-granting authority, and the rights client may instead be any variety of controlling authority that can also proxy-execute code. Accordingly, and more generally, in the present invention, a second process on a computer controls the operation and use of a first process on a computer by executing code on behalf of and as a proxy for the first process.

[0046] Turning now to Fig. 3, it is seen that in one embodiment of the present invention, a first process such as an application 30 is dependent upon a second

process such as a rights client 32 to proxy execute at least some portion of code for the application 30, where the rights client 32 includes a license evaluator 34 or the like. Accordingly, the rights client 32 may choose whether to in fact proxy execute the code for the application 30 based, among other things, on whether the license evaluator 34 has access to a license 36 corresponding to the application 30, and on whether the license 36 has permissions or rights that allow or at least do not prohibit the action corresponding to the code to be executed. Note that such a license 36 and the license evaluation 34 of the rights client 32 are known or should be apparent to the relevant public and therefore need not be disclosed herein in any detail.

[0047] In one embodiment of the present invention, and referring now to Fig. 4, the application 30, rights client 32, and license evaluator 34 are constructed to operate on a computer 110 (Fig. 1) or the like as follows. Typically, either a user or another process on the computer 110 instantiates the application 30 on such computer 110 as the aforementioned first process (step 401), and as part of an initializing process such application 30 ensures that the rights client 32 with the license evaluator 34 (hereinafter, 'rights client 32') is instantiated on the computer 110 as the aforementioned second process (step 403). Thereafter, the application 30 establishes a connection with the rights client 32 (step 405). Note that it may be the case that the rights client 32 is already instantiated or it may be the case the rights client 32 must be newly instantiated, either by the application 30, another process, the user, or the like.

[0048] Once step 405 is performed, and presuming that a license 36 corresponding to the application 390 is available to the rights client 32 and the license evaluator 34 thereof, the application 30 can query the rights client 32 to have the license evaluator 34 thereof determine based on the license 36 the rights the application 30 has based on such license 36, and the application 36 can then operate based on such rights. However, it is to be appreciated that a nefarious entity wishing to subvert the license 36 might choose to attack the application 30 by re-directing the query to a stub rights client that would in effect grant all rights to the application 30 without regard to any license 36, present or otherwise. Alternatively, such a nefarious entity might choose to spoof communications

between the application 30 and the rights client 32 or may wish to attack the rights client 32 itself if the application 30 cannot be attacked. Note, though, that the latter case is less likely inasmuch as the rights client 32 should be highly secure and protected from such an attack.

[0049] Accordingly, and in one embodiment of the present invention, the rights client 32 is required to proxy-execute at least some portions of code on behalf of the application 30 so that the application is dependent on the rights client 32. Put another way, by requiring the rights client 32 to proxy-execute at least some portion of code on behalf of the application 30, the aforementioned nefarious entity cannot subvert the license 36 by somehow removing the rights client 32 from participating in the method of Fig. 4. Instead, the rights client 32 must participate to proxy-execute code on behalf of the application 30, and while doing so the license evaluator 34 of the rights client 32 can also perform evaluation functions with regard to the license 36. Thus the rights client 32 does not merely provide the application 30 with a true or false type of response that could be spoofed.

[0050] In one embodiment of the present invention, the license 36 includes encoded information regarding the code that the rights client 32 is to proxy-execute. Thus, the license 36 must be available to the rights client 32 for same to proxy-execute on behalf of the application 30. For example, the encoded information may include the code, a reference to a location of the code, a decryption key for decrypting an encrypted version of the code, or the like.

[0051] As should now be appreciated, in order to effectuate proxy-execution, the application 30 must be pre-processed to define the code that is to be proxy-executed, to remove same from such application 30, and to appropriately store such removed code in a form proxy-executable by the rights client 32. In one embodiment of the present invention, then, and turning now to Fig. 5, a method of pre-processing the application 30 to effectuate proxy-execution is shown.

[0052] Preliminarily, and as may be appreciated, a developer develops source code 38 (Fig. 3) for the application 30 in an appropriate programming language, such as for example a C-type programming language (step 501). In doing so, and significantly, the developer identifies within such source code 38 for the application 30 each of one or more code sections that is to be proxy-executed

(step 503). As may be appreciated, each such proxy-executed code section identification may comprise any appropriate mark, tag, command, or the like without departing from the spirit and scope of the present invention. Thereafter, the developer compiles the source code 38 with a compiler 40 into machine code 42 (step 505).

[0053] Note that the developer may identify each code section within the source code 38 based on any particular criteria without departing from the spirit and scope of the present invention. For example, if the developer merely wishes to trigger proxy-execution from time to time so as to ensure the rights client 32 is present and is allowing the application 30 to operate based on a corresponding license 36, each such identified code section may be decided upon in a fairly random manner. However, if the developer wishes to trigger proxy-execution at specific times and/or with regard to specific sections of code, each such identified code section must be decided upon in a more targeted manner. Note with regard to the latter that it may be the case that an identified code section specifies a particular license right. In such a situation, it may also be the case that the rights client 32 will proxy-execute such identified code section only if the specified license right in the license 36 is met.

[0054] As may be appreciated, the compiler 40 may be any appropriate compiler without departing from the spirit and scope of the present invention. Significantly, the compiler 40 is constructed to maintain each code section identification in the machine code 42 so that post-compile processing may be performed on the code section identified thereby. Such maintaining maybe performed in any appropriate manner without departing from the spirit and scope of the present invention. For example, the compiler 40 may pass the identification from the source code 38 to the machine code 42 in a recognizable form, or may create a scratch table (not shown) with such information therein.

[0055] Thus, and in one embodiment of the present invention, after such compiling, the developer post-compiles the machine code 42 with each recognizable code section identification therein with a post-compiler 44 into the final code representative of the application 30, where the post-compiler 44 converts each identified code section into a form accessible only by the rights

client 32 and not by the application 30, such as for example by removing each identified code section in the machine code 42 from such application 30 or otherwise makes such identified code section inaccessible (step 507). As may be appreciated, such post-compiler 44 is constructed to retrieve each code section identification, either from the machine code 42, the aforementioned scratch table, or elsewhere, and operate based thereon.

[0056] In one embodiment of the present invention, for each identified code section in the machine code 42, the post-compiler 44 removes the identified code section from the machine code 42 (step 507a), replaces the removed code section with a triggering device (step 507b), notes an address of the triggering device within the application 30 (step 507c), and stores the removed code section and the noted address in a table 46 (Fig. 3) or the like (step 507e). If necessary or advisable, each removed code section may stored in the table 46 in an encrypted form decryptable by the rights client 32 (step 507d). As was set forth above, such table 46 may be made available to the rights client 32 by being set forth in the license 36, or by being set forth in another location. Note that the table 46 may be signed or otherwise protected from alteration by a verifying device such as a hash.

[0057] As may be appreciated, by replacing the removed code section with the triggering device, and presuming that the triggering device is shorter than the removed code section, the post-compiler 44 shortens the machine code 42. Note that the triggering device may be any appropriate triggering device without departing from the spirit and scope of the present invention, as long as the triggering device is recognizable as a signal that the rights client 32 is needed to proxy-execute the corresponding removed code section. For example the triggering device may be a particular exception that would get the attention of the rights client 32.

[0058] After the post-compiler 44 is finished, and as should now be appreciated, such post-compiler 44 outputs final code representative of the application 30 (hereinafter, 'the application 30') and the table 46 (step 509). As was set forth above, such table 46 may be made available to the rights client 32 by being set forth in the license 36, or by being set forth in another location separate from the application 30. It may for example be the case that the table 46 with encrypted

removed code sections therein is placed in the license 36 along with a decryption key for decrypting each encrypted code section, where the decryption key is itself encrypted in a manner decryptable by the rights client 32. Note that by separating the table 46 from the application 30, the application 30 has no innate access to the table 46 or the removed code sections therein.

[0059] Thus, and returning now to Fig. 4, during runtime, and after the application 30 and rights client 32 have been instantiated, the rights client 32 attaches itself to the application 30 in the manner of a debugger or the like so that the rights client 32 can monitor the application 30 for when each triggering device / exception therein is executed (step 407). As may be appreciated, the rights client 32 monitors the application 30 for the particular triggering device / exception (hereinafter, 'exception') that signals that the rights client 32 is to proxy-execute on behalf of the application 30. Thus, on every breakpoint exception, the rights client 32 determines whether the exception source is a removed code section, and if so the rights client proxy-executes the removed code section, presuming the license 36 so allows.

[0060] In an alternate embodiment of the present invention, the rights client 32 does not attach itself to the application 30 to monitor for an exception, but instead receives the exception from an operating system operating the computer 110. However, such an arrangement is indirect and therefore slower. Another alternative would be to have each triggering device be a call to the rights client 32, although such a strategy is slightly more complex as compared to an exception and is more prone to attack by a nefarious entity.

[0061] At some point, the application 30 may explicitly request permission to operate from the rights client 32 based on the license 36. In response, the rights client 32 searches for the license 36, the license evaluator 34 evaluates such license 36, and the rights client 32 returns such requested permission if the evaluation of the license evaluator 34 is positive. Note, though, that such explicit request for permission and response are ancillary to the present invention. Rather, in the present invention, the rights client 32 is actuated based on an exception or the like from the application 30 and not based on an explicit request from the application 30. Thus, in the present invention, the rights client 32 can

withhold performance of a function on behalf of the application 30 even when the application 30 never requested permission to perform such function.

[0062] At any rate, in the course of operating, the application 30 at some point executes an exception in the code thereof, where such exception was placed in the application 30 by the post-compiler 44 in place of a removed identified code portion (step 409). As should be understood, upon executing the exception, the application 30 halts until receiving notice that the exception has been dealt with (step 411). Inasmuch as the rights client 32 is attached to the operating application 30 and is listening for such exception from such application 30, such rights client 32 notes the exception (step 413) and responds thereto (step 415).

[0063] In particular, to respond to the exception, the rights client first determines the address of the exception within the application 30 (step 415a), locates the corresponding code section in the table 46 based on such address (step 415b), proxy-executes such corresponding code section on behalf of the application (step 415e), and then signals to the application 30 that the exception has been dealt with (step 415f). As may be appreciated, the application 30 may then proceed (step 417). Note that if the corresponding code section is encrypted, the rights client 32 must decrypt the located corresponding code section before proxy-executing same (step 415c). Note, too, that a particular code section may require that the license evaluator 34 of the rights client 32 first verify that the license 36 grants the rights necessary to proxy-execute such code section on behalf of the application 30 (step 415d). As may be appreciated, the rights client 32 proxy-executes such code section only if the license grants the right to do so. Otherwise, the rights client 32 declines to do so. In the latter case, it may be that the rights client 32 returns an appropriate message to the application 30.

[0064] It is to be appreciated that a rights client 32 should not be proxy-executing any arbitrary code section, especially inasmuch as the rights client 32 should be especially secure and therefore could have a relatively large amount of operating rights with respect to the computer 110. Put another way, the rights client 32 should not be performing actions that the application would not have operating rights to perform, such as altering certain system registers, accessing memory areas of other applications and the operating system, and the like.

Accordingly, in one embodiment of the present invention, the post-compiler 44 during operation thereof ensures that each code section removed and stored thereby is not of a sensitive nature. For example, it may be the case that the post-compiler 44 during operation thereof ensures that each such code section does not affect system memory. Of course, other bases for filtering code sections may be employed without departing from the spirit and scope of the present invention. Note that if a code section includes sensitive code, it may be that the post-compiler isolates such sensitive code and removes only sub-portions of code on either side of the sensitive code.

[0065] In one embodiment of the present invention, the rights client 32 proxy-executes on behalf of the application only if a valid license 36 corresponding to such application 30 is available to the rights client 32. In such a case, it may be that the purpose of each exception and proxy-execution based thereon is merely to occasionally check that the license 36 is still present and still valid. In an alternate embodiment, the rights client 32 proxy-executes on behalf of the application without regard to any corresponding valid license 36. In such a case, it may be that the purpose of each exception and proxy-execution based thereon is merely to tie the application 30 to the rights client 32, which presumably is tied to the computer 110, thus tying the application 30 to the computer 110.

[0066] In one embodiment of the present invention, the application 30 as produced by the post-compiler 44 may include multiple types of exceptions, each triggering the rights client 32. However, each different type of exception is handled differently. For one example, one type of exception may require the rights client 32 to check the license 36 while another type may not. For another example, different types of exceptions could require access to different tables 46, or could require different decryption keys and/or methods.

[0067] As disclosed herein, the application 30, the rights client 32, and the license 36 are separate constructs. Nevertheless, it should be appreciated that such items may be combined in any manner without departing from the spirit and scope of the present invention. For example, the application 30 could include the rights client 32, or the rights client 32 could include the license 36. Note, though,

that in at least some instances combined items may be more susceptible to an attack from a nefarious entity.

[0068] As also disclosed herein, the rights client 32 proxy-executes code on behalf of the application 30. Alternatively, the rights client 32 may operate to modify the application 30 to include the to-be-executed code, allow such application to execute such code, and then again modify the application 30 to remove such code. Note, though, that such an arrangement may be more susceptible to attack by a nefarious entity, especially in the moments when the application 30 is modified to include the to-be-executed code.

[0069] As may be appreciated, one especially useful aspect of the present invention is that the rights client 32 may now perform especially secure functions on behalf of the application 30 such that a nefarious entity is thwarted from affecting such functions. For example, it may be the case that a term in a license 36 affects how many times the application 30 can perform a specific action. Although the application 30 could obtain such term from such license 36, having the application 30 do so could allow a nefarious entity to intervene in the process to subvert same. Instead, in one embodiment of the present invention, the rights client 32 is employed to proxy-execute code for the application 30 relating to such term in such license 36, including obtaining the term and employing same.

Conclusion

[0070] The programming necessary to effectuate the processes performed in connection with the present invention is relatively straight-forward and should be apparent to the relevant programming public. Accordingly, such programming is not attached hereto. Any particular programming, then, may be employed to effectuate the present invention without departing from the spirit and scope thereof.

[0071] In the present invention, a method and mechanism are provided by which a rights client 32 with a license evaluator 34 control operation and use of an application 30 based on a license 36 corresponding thereto. The rights client 32 with the license evaluator 34 executes certain portions of code on behalf of and as

a proxy for the application 30, where the license evaluator 34 can determine if the license 36 allows such execution.

[0072] It should be appreciated that changes could be made to the embodiments described above without departing from the inventive concepts thereof. It should be understood, therefore, that this invention is not limited to the particular embodiments disclosed, but it is intended to cover modifications within the spirit and scope of the present invention as defined by the appended claims.